AMENDMENTS TO THE SPECIFICATION

Page 9, please amend the paragraph beginning on line 19 as follows:

Following the step of sorting and arranging **208**, a type array is created by extracting types found in the second field of a typing tuple according to the sorted order of ambiguity typing sequences in step **210**. Types not included in ambiguity sequences, which are also extracted from the second field of typing tuples, are listed following those typing tuples that are members of an ambiguity typing sequence. It is of note that multiple entries for a given type may exist if the type is included in multiple ambiguity sequences. Entries in a type array that correspond to type names with an offset number assigned as described previously, are also given the same offset number. Those entries that have no offset number are assigned an offset number of zero. For each type record node N **212**, if the intersection of typing tuples in N with any ambiguity typing sequences is non-empty **214**, then replace first typing tuple in N with typing tuple having offset **216**. As a last step **218** in the algorithm of the present invention, an index structure is created to link each type name extracted from a first field of a typing tuple to its corresponding type. The process terminates in step **220**. Index entries will have a string field denoting element type, a flag field denoting ambiguity, and an index field denoting the index of an element type in a type array. A flag field is given a value of 'Y' if a corresponding element type is ambiguous and 'N' if it is not ambiguous. An index field is given a value corresponding to the index of an element type in a type array if a corresponding flag field is set to 'N' and the first index entry in a type array for an ambiguity sequence if a corresponding flag field is set to 'Y'. An index structure is implemented by, but is not limited to, one of the following data structures: hash tables, binary trees, and B+ trees.

Page 10, please amend the paragraph beginning on line 17 as follows:

The exemplary XML schema in Figure 3 comprises such features as an abstract element type-~~304~~, complex type-~~306~~, anonymous element type-**312**, and substitution group-~~316~~. Figure 3 is used to illustrate the execution of a first XML compilation algorithm of the present invention.

Page 10, please amend the paragraph beginning on line 21 as follows:

Figure 4 illustrates a type hierarchy tree built by an initial step of a first algorithm of the present invention for the XML schema shown in Figure 3. Except for "namespace:p" root node **400**, all nodes are type records. Determined from a type hierarchy tree of nodes **402**, **404**, **406**, **408**, **410**, **412**, **414**, and **416** is the following typing set.

Page 13, please amend the paragraph beginning on line 18 as follows:

In an initial step **600** of the annotate_type algorithm, type annotation records from precompiled data structures shown in Figure 5 are loaded into memory. An empty offset stack is then created, and a value of zero is pushed onto an empty offset stack. While there is an XML document or document fragment remaining to be annotated **602**, it is determined **604** whether tuples comprising the following combinations are encountered; <start tag, element_name> **606**, <start tag, "element name", xsi:type="type name" > **608**, <attribute, "attribute name"> **610**, and <end tag> **612**. If a tuple comprising a start tag and an element name (e.g., <start tag, element_name>) is encountered as in step **606**; then in step **614** type indexing data structure **500** is searched with respect to element name **502** to determine an index **504**. Also occurring in step **614**, if index **504** determined has a positive indication **506** (e.g., 'Y' as shown in Figure 5), then

an index **504** is incremented by a PEEK value of the offset stack. A PEEK value of an offset stack is used to determine the value of the entry on the top of an offset stack. An index **504** is incremented in order to add the index from an index structure with the offset determined by a PEEK value. The resultant index of a given element type in a typing array is used to annotate an element. Also in step **614**, the element is then annotated with type **510** stored in typing array **508** at index location **504** determined by previous searching step. Lastly in step **614**, a record containing the offset **512** stored in a typing array **508** at an index location **504** determined by a previous searching step is pushed onto an offset stack. The same process is followed in step **616** if a tuple comprising a start tag, element name, type, and type name (e.g., <start tag, "element name", xsi:type="type name">) is encountered in step **608**; except a type indexing data structure **500** is searched with respect to type name **502** rather than an element name to determine an index **504**. The same process is also followed in step **618** if an attribute and attribute name tuple are encountered **610** (e.g., <attribute, "attribute name">); however, a type indexing data structure **500** is searched with respect to attribute name **502** to determine an index **504**. In addition, a record is not pushed onto the offset stack. Lastly, if an end tag is encountered as in step **612**, the top record in the offset stack is popped off in step **620**. The process terminates in step **622**.